1. A history is *serializable* if its effects on the database are the same as that of some serial history, regardless of what the initial state of the database is. Consider each of the following histories:

(a) \( r_1[A], r_2[A], r_3[B], c_3, w_1[A], r_2[C], w_2[B], c_2, w_1[C], c_1 \)
(b) \( r_1[A], r_2[A], w_1[B], w_2[B], r_1[B], r_2[B], w_2[C], c_2, w_1[D], c_1 \)

Answer the following questions about each of the above histories.

(a) Is the history serializable according to the definition given above?
(b) Draw the serialization graph for the history.
(c) Is the history conflict-serializable? If so, what are all the equivalent serial histories?
(d) Is the history recoverable?
(e) Does the history avoid cascading aborts?
(f) Is the history strict?

2. The action of multiplication by a constant factor can be modeled by an action of its own. Suppose \( MC(X, c) \) stands for an atomic execution of the steps \( READ(X, t); t := c * t; WRITE(X, t) \); where \( X \) is a data item and \( t \) is a temporary variable used by the transaction. We can also introduce a lock mode that allows only multiplication by a constant factor. Show the compatibility matrix for read, write, and multiplication-by-a-constant lock.

3. Here is a history with one action missing:

\( r_1[A], r_2[B], \ldots \ldots, w_1[C], w_2[A] \)
Your problem is to figure out what actions of certain types could replace the non-serializable operations and make the history not be conflict serializable. Tell all possible non-serializable replacements for each of the following types of operation:

(a) Read operation
(b) Write operation

4. Consider the following history:

\[ r_1[A], r_2[B], r_3[C], r_1[B], r_2[C], r_3[D], w_1[A], w_2[B], w_3[C] \]

Do each of the following:

(a) Insert read and write locks, and insert unlock actions. Place a read lock immediately in front of each read operation that is not followed by a write operation of the same data item by the same transaction. Place a write lock in front of every other read or write operation. Place the necessary unlocks at the end of every transaction.

(b) Tell what happens when each history is run by a scheduler that supports read and write locks.

5. Are two histories having identical serialization graph conflict equivalent? Explain.

6. Two transactions are not interleaved in a history if every operation of one transaction precedes every operation of the other. Give an example of a serializable history \( H \) that has all of the following properties:

(a) transactions \( T_1 \) and \( T_2 \) are not interleaved in \( H \),
(b) \( T_1 \) precedes \( T_2 \) in \( H \), and
(c) in any serial history equivalent to \( H \), \( T_2 \) precedes \( T_1 \)

7. Consider the following history \( H \)

\[ r_2[y], r_1[x], r_3[y], r_2[x], w_2[y], c_2, w_1[x], c_1, r_3[x], c_3 \]

Assuming that each transaction is consistent, does the final database state satisfy all the integrity constraints? Explain.

8. Suppose that transactions \( T_1 \) and \( T_2 \) can be decomposed into the subtransactions \( T_1 : T_{11}, T_{12} \) and \( T_2 : T_{21}, T_{22} \) such that each subtransaction individually maintains the integrity constraints of the database. Instead of guaranteeing that all histories involving \( T_1 \) and \( T_2 \) are conflict-serializable, suppose that the concurrency control guarantees that all subtransactions are executed using 2PL.
(a) Will we always get a conflict-serializable execution of \( T_1 \) and \( T_2 \)? Explain.

(b) Will integrity constraints be maintained by all possible histories? Explain.